Stochastic optimization of energy systems on high-performance computers

Cosmin G. Petra

Mathematics and Computer Science Division

Argonne National Laboratory

petra@mcs.anl.gov

INFORMS COMPUTING SOCIETY CONFERENCE 2015

Richmond, Virginia

Jan 11, 2015

Outline

- Parallel interior-point solver for stochastic optimization - PIPS-IPM
  - Algorithmic developments
  - Implementation
  - Performance studies: parallel efficiency and time-to-solution
- Modeling frameworks - StochJuMP
- Application: the impact of wind correlation on power grid economic dispatch operations

Stochastic two-stage problems with recourse

- Traditionally known as two-stage stochastic programming with recourse
- The first decision stage is deterministic and corresponds to the “now” decision
- The second decision stage depends on the random event and the first-stage decision; it gives event-dependent decisions.
- The second-stage decisions are **recourse actions** that minimize the “bad” effects caused by the first-stage decision.
  - In other words, the second-stage consists of a minimization problem

$$\underset{x_0}{Min}\left\{ g_0(x_0) + \mathbb{E}\left[\underset{x}{Min}\, g(x,\xi) \right] \right\}$$

subj. to. $W(\xi)x = b(\xi) - T(\xi)x_0$

$x \geq 0$

subj. to. $A_0 x_0 = b_0$

$x_0 \geq 0$

- The objective is to minimize the cost associated with the first-stage decisions plus the expectation of the recourse cost.

Optimization under uncertainty

- Two-stage stochastic programming with recourse ("here-and-now")

$$\min_{x_0} \left\{ f_0(x_0) + \mathbb{E}\left[\min_{x} f(x,\omega) \right] \right\}$$

$$\text{subj. to. } W(\omega)x = b(\omega) - T(\omega)x_0$$

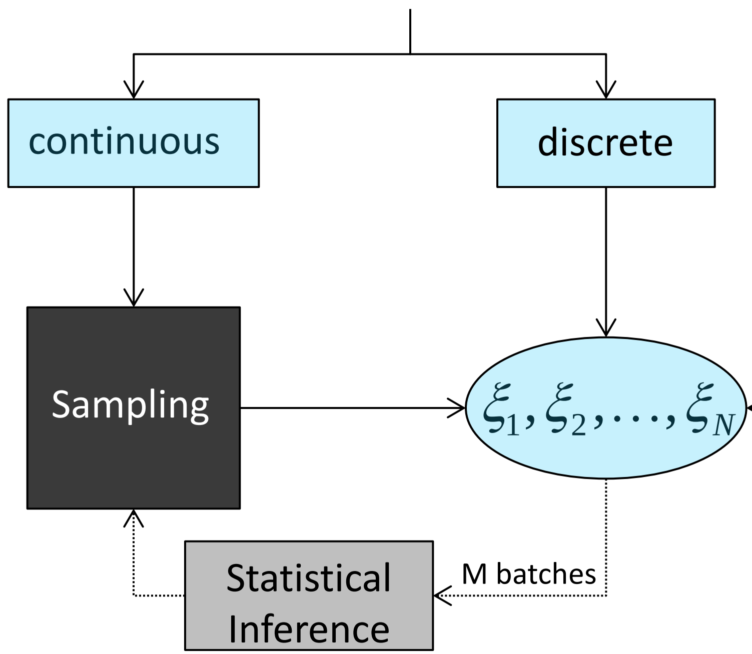
$$x \geq 0$$

$$\text{subj. to. } A_0 x_0 = b_0$$

$$x_0 \geq 0$$

- $\xi(\omega) := (W(\omega), T(\omega), b(\omega), Q(\omega), c(\omega))$

$$f(x,\omega) = \frac{1}{2} x^T Q(\omega) x + x^T c(\omega)$$



Sample average approximation (SAA)

$$\begin{aligned}
\min_{x_0, x_1, x_2, \ldots, x_N} \quad & f_0(x_0) + \frac{1}{N}\sum_{i=1}^{N} f_i(x_i) \\
\text{subj. to.} \quad & A_0 x_0 = b_0 \\
& T_k x_0 + W_k x_k = b_k, \\
& x_0 \geq 0, \quad x_k \geq 0, \quad k = 1, \ldots, N.
\end{aligned}$$

Large-scale (dual) block-angular LPs

Extensive form

$$
\begin{array}{lcccccccccccc}
\min & c_0^T x_0 & + & c_1^T x_1 & + & c_2^T x_2 & + & \dots & + & c_N^T x_N & & \\
\text{s.t.} & A x_0 & & & & & & & & & = & b_0, \\
& T_1 x_0 & + & W_1 x_1 & & & & & & & = & b_1, \\
& T_2 x_0 & & & + & W_2 x_2 & & & & & = & b_2, \\
& \vdots & & & & & & \ddots & & & & \vdots \\
& T_N x_0 & & & & & & & + & W_N x_N & = & b_N, \\
& \multicolumn{11}{l}{x_0 \geq 0, \quad x_1 \geq 0, \quad x_2 \geq 0, \quad \dots, \quad x_N \geq 0.}
\end{array}
$$

- In terminology of stochastic LPs:
  - First-stage variables (decision now): x_0
  - Second-stage variables (recourse decision): $x_1, \dots, x_N$
  - Each diagonal block is a realization of a random variable (scenario)

Computational challenges and difficulties

- May require many scenarios (100s, 1,000s, 10,000s …) to accurately model uncertainty
- "Large" scenarios (W_i up to 250,000 x 250,000)
- "Large" 1st stage (1,000s, 10,000s of variables)
- Easy to build a practical instance that requires **terabytes** of RAM to solve

  ➔ Requires distributed memory

- Current practice in power grid industry is to solve 24-hour horizon (deterministic) instances in under 1 hour

- Need to solve under strict time requirements

Solving the SAA problem

- Interior-point methods (IPMs) applied to the extensive form
  - Polynomial iteration complexity: $O(\sqrt{n}L)$ (in theory, but is conservative)
  - IPMs perform better in practice (infeasible primal-dual path-following)
  - Two linear systems solved at each iteration
  - Direct solvers needs to be used because IPMs linear systems are ill-conditioned and needs to be solved accurately
  - We solve the SAA problems with a *standard* IPM (Mehrotra's predictor-corrector) and *specialized* linear algebra → **PIPS-IPM** (Petra et.al.)
- Alternative algorithms: Benders-type decompositions, simplex, stochastic gradient
  - Much easier to implement but higher iteration complexity (not scalable)
  - Also inferior convergence (local) behavior (not accurate)
  - Ratio of computations and communication lower -> latency affects parallel efficiency
  - Prone to load imbalancing
  - Suitable for heterogeneous computing platforms (see ATR: Wright, Linderoth, 2005)

Linear algebra of primal-dual interior-point methods (IPM)

Convex quadratic problem

$$\text{Min } \frac{1}{2} x^T Q x + c^T x$$

$$\text{subj. to. } Ax = b$$

$$x \geq 0$$

IPM Linear System

$$\begin{bmatrix} Q+\Lambda & A^T \\ A & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = rhs$$

Multi-stage SP

Two-stage SP

nested arrow-shaped linear system (modulo a permutation)

$$\begin{bmatrix} Q_1 & W_1^T & & & & & & 0 & 0 \\ W_1 & 0 & & & & & & T_1 & 0 \\ & & Q_2 & B_2^T & & & & 0 & 0 \\ & & W_2 & 0 & & & & T_2 & 0 \\ & & & & \ddots & & & \vdots & \vdots \\ & & & & & Q_S & W_S^T & 0 & 0 \\ & & & & & W_S & 0 & T_S & 0 \\ 0 & T_1^T & 0 & T_2^T & \dots & 0 & T_S^T & Q_0 & A_0^T \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & A_0 & 0 \end{bmatrix}$$

Special Structure of KKT System (Arrow-shaped)

$$\begin{bmatrix} K_1 & & & B_1 \\ & \ddots & & \vdots \\ & & K_N & B_N \\ B_1^T & \dots & B_N^T & K_0 \end{bmatrix} \begin{bmatrix} \Delta z_1 \\ \vdots \\ \Delta z_N \\ \Delta z_0 \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_N \\ r_0 \end{bmatrix}$$

where,

$$K_i := \begin{bmatrix} \bar{Q}_i & W_i^T \\ W_i & 0 \end{bmatrix}, \qquad K_0 := \begin{bmatrix} \bar{Q} & A^T \\ A & 0 \end{bmatrix},$$

$$B_i := \begin{bmatrix} 0 & 0 \\ T_i & 0 \end{bmatrix}, \qquad i = 1, 2, \dots, N.$$

Block Elimination

$$\begin{bmatrix} K_1 & & & B_1 \\ & \ddots & & \vdots \\ & & K_N & B_N \\ B_1^T & \dots & B_N^T & K_0 \end{bmatrix} \begin{bmatrix} \Delta z_1 \\ \vdots \\ \Delta z_N \\ \Delta z_0 \end{bmatrix} = \begin{bmatrix} r_1 \\ \vdots \\ r_N \\ r_0 \end{bmatrix}$$

Multiply row *i* by $-B_i^T K_i^{-1}$ and sum all the rows to obtain

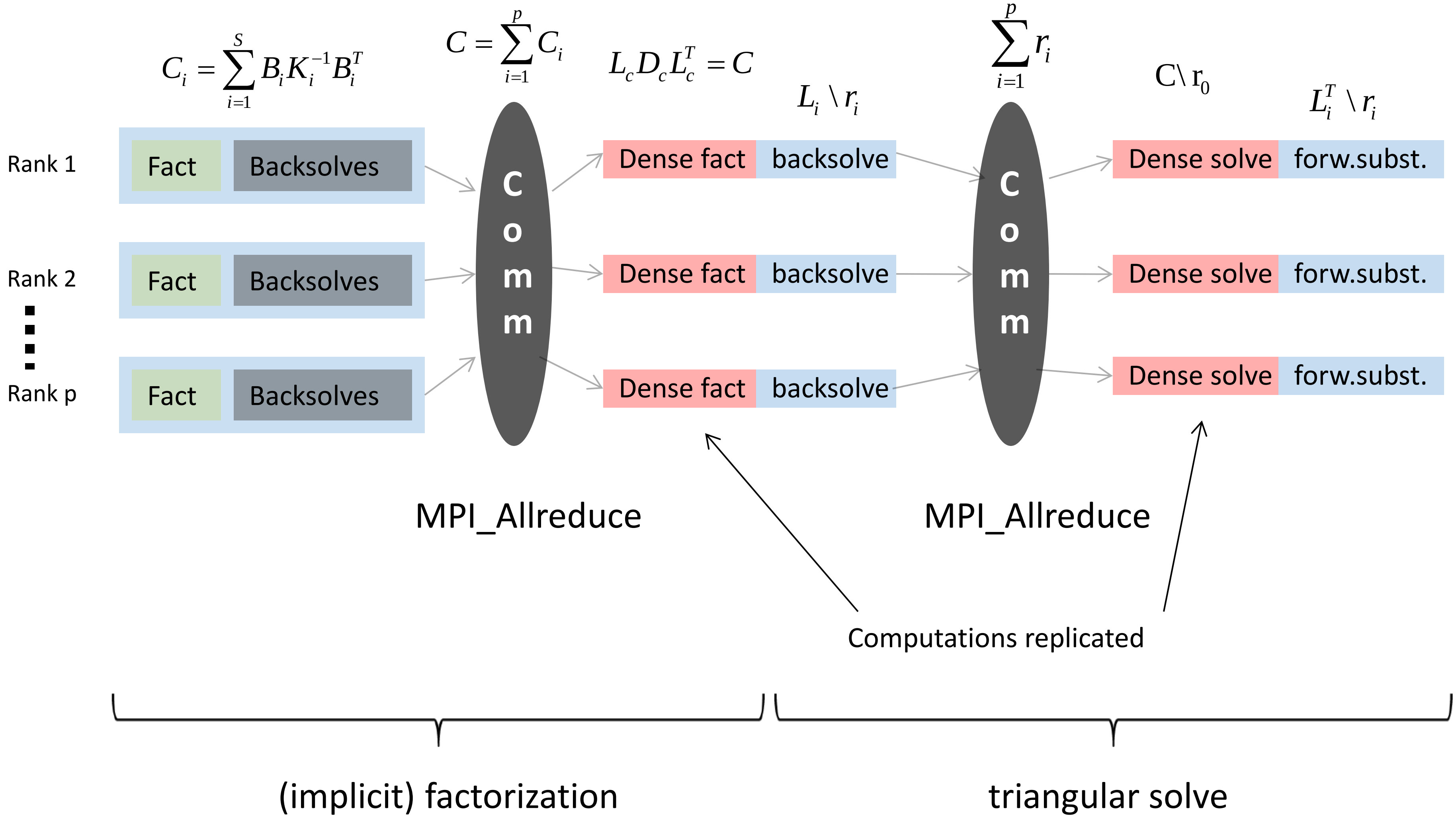
$$\left(K_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} B_i \right) \Delta z_0 = r_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} r_i$$

The matrix $C := K_0 - \sum_{i=1}^{N} B_i^T K_i^{-1} B_i$ is the Schur-complement of the diagonal $K_1, \dots, K_N$ block.

Solution Procedure for KKT System – a compact view

1. Calculate $B_i^T K_i^{-1} B_i,\ i = 1, \ldots, N$ ("Compute S.C.")
2. Form $C := K_0 - \sum_{i=1}^N B_i^T K_i^{-1} B_i$ ("Form S.C.")
3. Factorize $C = L_0 D_0 L_0^T$ ("Factor S.C.")
4. Solve $\Delta z_0 = C^{-1}(r_0 - \sum_{i=1}^N B_i^T K_i^{-1} r_i)$
5. Solve $\Delta z_i = K_i^{-1}(B_i \Delta z_0 - r_i),\ i = 1, \ldots, N$

Parallel computational pattern



Implementation considerations

- Codename: **PIPS-IPM**
- C++ code based on OOQP optimization solver (S. Wright & M. Gertz, ANL 2003)
- Hybrid parallel: MPI+OpenMP/GPU.

- Data matrices are **sparse**

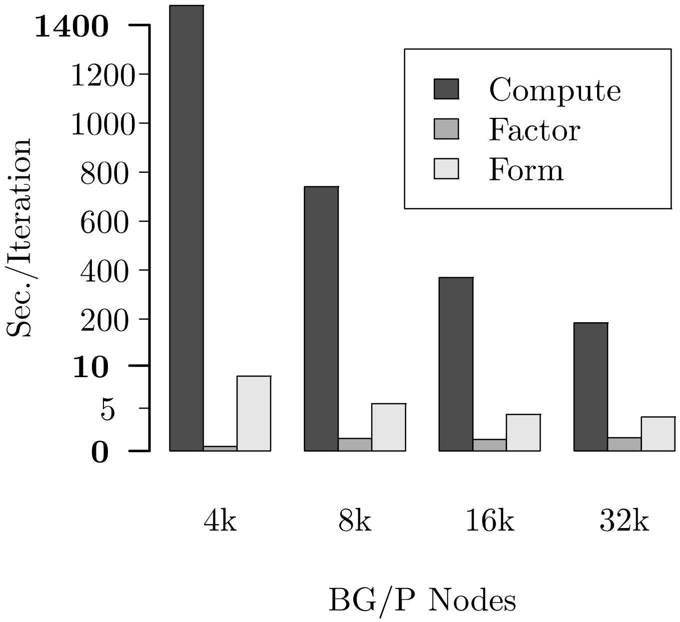
- Direct (sparse and dense) factorizations are needed
  - **saddle-point** linear systems: symmetric but indefinite
  - increasingly **ill-conditioned** as the optimality is approached.

- Second-stage linear systems handled with off-the-shelf **sparse** linear solvers (MA27/57/86 or PARDISO, other can be adopted as well)

- The **dense** Schur complement is solved using LAPACK/MAGMA.

Strong scaling – BG/P

# Nodes	Scenarios per Node	Execution Time (min.)	Scaling Efficiency
4,096	8	125.	—
8,192	4	63.	99%
16,384	2	32.	98%
32,768	1	16.	96%



For real-time simulations "Compute" times need to be drastically reduced!

Incomplete augmented factorization

"Compute S.C." (Step 1): $L_{22}U_{22} = -B_i^T K_i^{-1} B_i$

$$\begin{bmatrix} K_i & B_i^T \\ B_i & 0 \end{bmatrix} = \begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix}$$

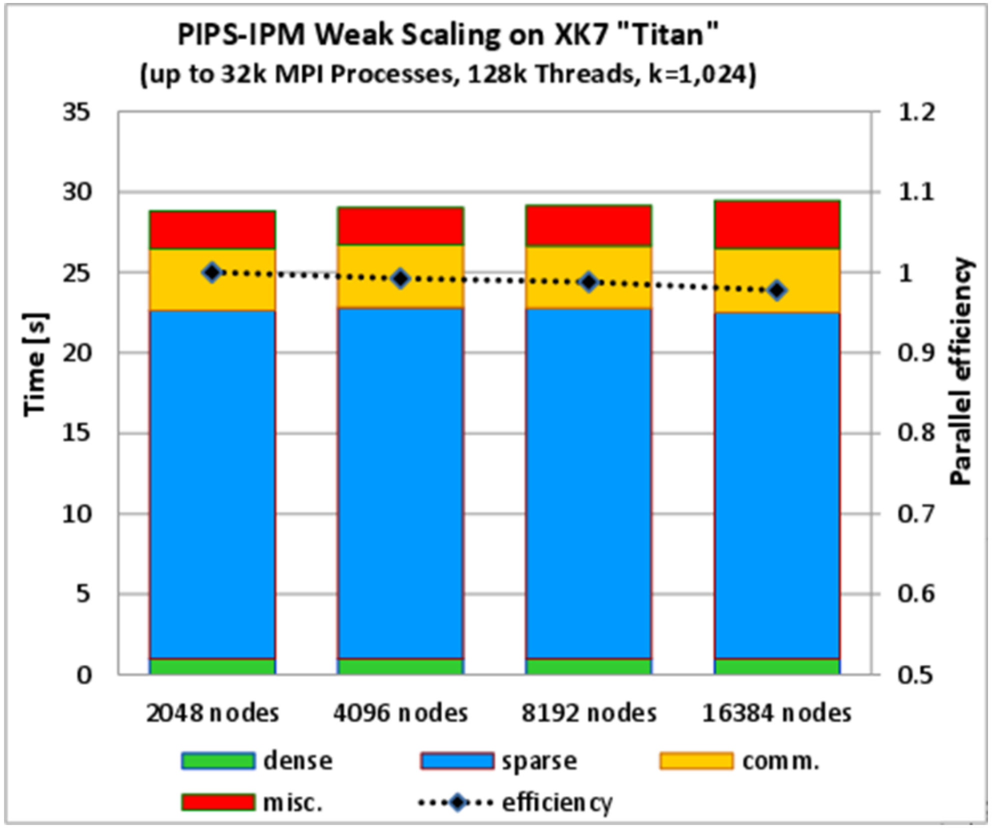
UC12 – BG/P

Test Problem	Solver	Threads/MPI	$B_i^T K_i^{-1} B$ Time (Sec.)
UC4	MA57	1	29.65
	PARDISO-SC	1	2.45
	PARDISO-SC	2	1.32
	PARDISO-SC	4	0.79
UC12	MA57	1	292.02
	PARDISO-SC	1	50.39
	PARDISO-SC	2	26.09
	PARDISO-SC	4	13.81
UC24	MA57	1	>3600
	PARDISO-SC	1	308.193
	PARDISO-SC	2	157.63
	PARDISO-SC	4	81.11

UC24 – XC30

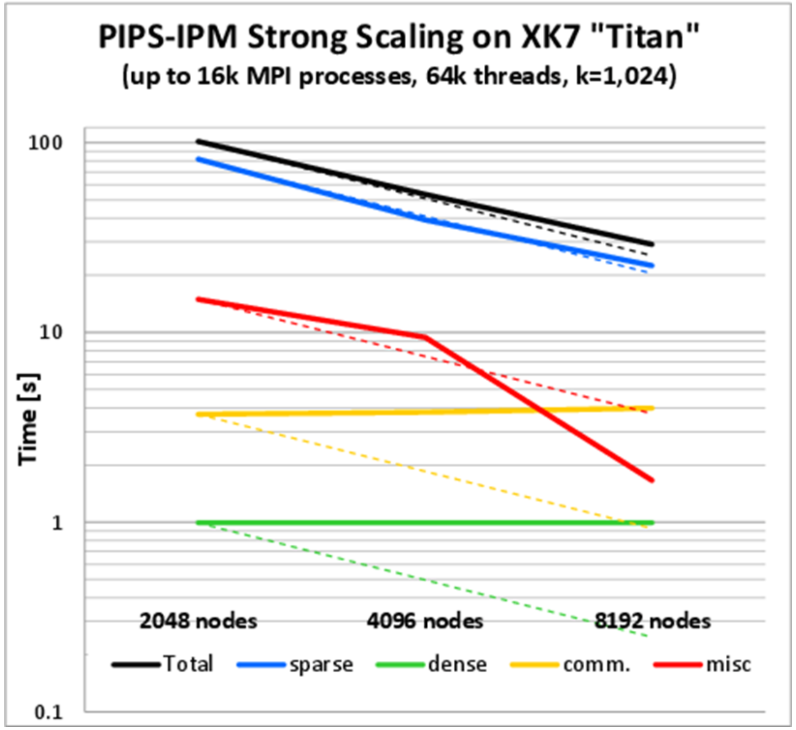
# of MPI ranks	(b) Number of threads per process on XC30					
	1	1	2	4	8	16
	PARDISO	PARDISO-SC				
1	168.3	19.0	12.9	8.4	6.2	5.9
2	175.3	18.9	12.4	8.5	7.4	
4	194.6	19.5	13.8	12.1		
8	284.2	22.1	21.8			
16	281.6	38.5				

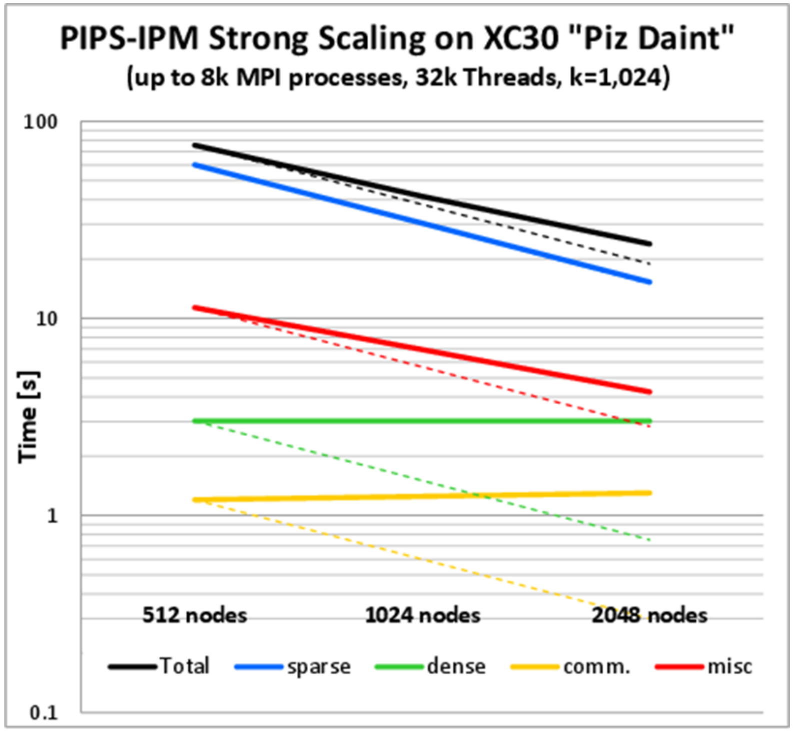
Weak scaling efficiency



Largest instance has 4.08 billion decision variables and 4.12 billion constraints.

Strong scaling





The instance used in the XK7 runs has 4.08 billion decision variables and 4.12 billion constraints.

Solve to completion – UC12 on BG/P

- Before: 4 hours 10 minutes wall time to solve UC4 problem with 8k scenarios on 8k nodes (on BG/P)
- Now: UC24 – 32,768 scenarios in approx. 40 minutes
- UC12 on BG/P

Nodes/scens	Wall time (sec)	IPM Iterations	Time per IPM iteration (sec)
4096	3548.5	103	33.57
8192	3883.7	112	34.67
16384	4208.8	123	34.80
32768	4781.7	133	35.95

- In addition to implementation, **the algorithm is also scalable**

StochJuMP - parallel algebraic modelling for stochastic optimization

J. Huchette, M. Lubin, C. Petra , “*Parallel algebraic modeling for stochastic optimization,*” High Performance Technical Computing in Dynamic Languages (HPTCDL), SC’14.

Expressing and constructing the stochastic optimization problem

- Express the problem in a human-readible, mathematical format
- Automatic transformation to the low-level format of the solver(s)
  - **efficient and distributed-memory generation of the large models**

$$
\begin{array}{rccccccccccc}
\min & c_0^T x_0 & + & c_1^T x_1 & + & c_2^T x_2 & + & \dots & + & c_N^T x_N & & \\
\text{s.t.} & A x_0 & & & & & & & & & = & b_0, \\
 & T_1 x_0 & + & W_1 x_1 & & & & & & & = & b_1, \\
 & T_2 x_0 & & & + & W_2 x_2 & & & & & = & b_2, \\
 & \vdots & & & & & & \ddots & & & & \vdots \\
 & T_N x_0 & & & & & & & + & W_N x_N & = & b_N, \\
 & \multicolumn{11}{l}{x_0 \geq 0, \quad x_1 \geq 0, \quad x_2 \geq 0, \quad \dots, \quad x_N \geq 0.}
\end{array}
$$

- The problem's structure is passed **transparently** to the solver

JuMP – modeling language for Mathematical Programming in Julia

- Miles Lubin (MIT), Iain Dunning (MIT)
- Julia – a fresh approach to scientific and technical computing
  - high-level, high-performance, open-source dynamic language for technical computing
  - keeps productivity of dynamic languages without giving up speed (2x of C/C++/Fortran)
- JuMP - compact, easy-to-use AML in Julia for modelling LP/QP/MILP/MIQCQP

```
m = Model(:Max)
@defVar(m, 0 <= x[j=1:N] <= 1)
@setObjective(m, sum{profit[j] * x[j], j=1:N})
@addConstraint(m, sum{weight[j] * x[j], j = 1:N} <= C)
```

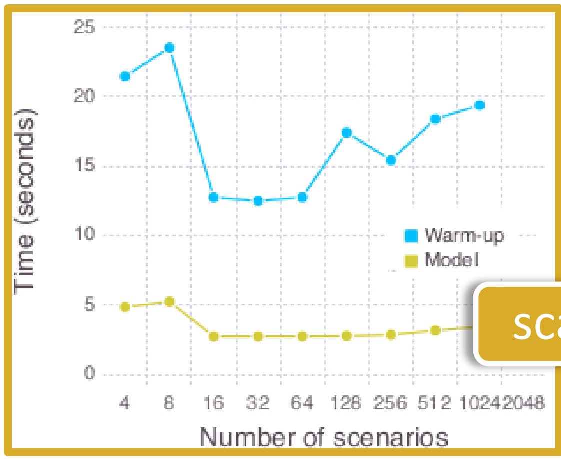
- Macros instead of operator overloading (known to have poor performance)
- Efficient sparse internal generation and representation of the data
- Nonlinear programming fully functional

Table: Linear-quadratic control benchmark results. N=M is the grid size. Total time (in seconds) to process the model definition and produce the output file in LP and MPS formats (as available).

	JuMP/Julia		AMPL	Gurobi/C++		Pulp/PyPy		Pyomo
N	LP	MPS	MPS	LP	MPS	LP	MPS	LP
250	0.5	0.9	0.8	1.2	1.1	8.3	7.2	13.3
500	2.0	3.6	3.0	4.5	4.4	27.6	24.4	53.4
750	5.0	8.4	6.7	10.2	10.1	61.0	54.5	121.0
1,000	9.2	15.5	11.6	17.6	17.3	108.2	97.5	214.7

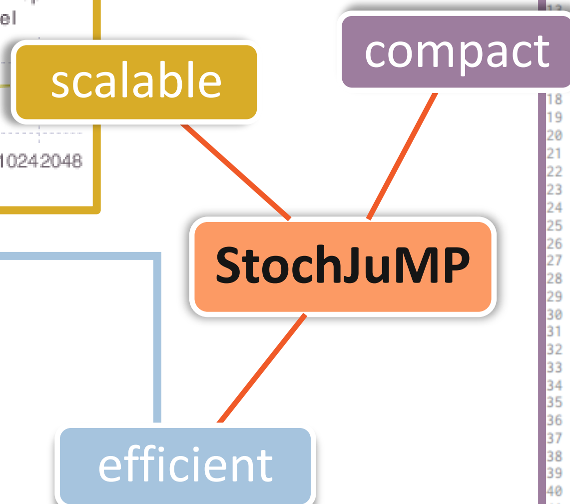
StochJuMP - parallel algebraic modelling for stochastic optimization (Huchette, Lubin, Petra -2014)

- Extension of JuMP for stochastic LP/QP/MILP/ MIQCQP
- Interfaced with PIPS, runs efficiently on “Blues” LCRC cluster
- Parallel, memory-distributed generation of the model



N	Load Module	Generate Model
4	3.528	1.409
8	3.694	1.415
16	3.334	1.169
32	2.541	0.917
64	2.863	0.996
128	3.462	0.768
256	2.620	0.723
512	2.871	0.689
1024	1.470	0.384
2048	-	0.500

RATIO OF STOCHJUMP TIMINGS OVER SOLVE TIME ($\times 100$).




```julia
m = StochasticModel(NS)

# Stage 0
@defVar(m, 0 <= Pgen_f[i=GENTHE] <= np_capThe[i])
@defVar(m, 0 <= PgenWin_f[i=GENWIN] <= np_capWin[i])
@defVar(m, -lineCutoff*Pmax[i] <= P_f[i=LIN] <= lineCutoff*Pmax[i])

# (forward) power flow equations
@addConstraint(m, pfeq_f[j=BUS],
               +sum{P_f[i], i=LIN; j==rec_bus[i]}
               -sum{P_f[i], i=LIN; j==snd_bus[i]}
               +sum{Pgen_f[i], i=GENTHE; j==bus_genThe[i]}
               +sum{PgenWin_f[i], i=GENWIN; j==bus_genWin[i]}
               -sum{loads[i], i=LOAD; j==bus_load[i]} >= 0)

@second_stage m node begin
    bl = StochasticBlock(m)
    # variables
    @defVar(bl, 0 <= Pgen[i=GENTHE] <= np_capThe[i])
    @defVar(bl, 0 <= PgenWin[i=GENWIN] <= windPower[node,i])
    @defVar(bl, -lineCutoff*Pmax[i] <= P[i=LIN] <= lineCutoff*Pmax[i])
    @addConstraint(bl, rampUpDown[g=GENTHE],
                   -0.1np_capThe[g] <= Pgen[g] - Pgen_f[g] <= 0.1np_capThe[g])
    # (spot) power flow equations
    @addConstraint(bl, pfeq[j=BUS],
                   +sum{P[i]-P_f[i], i=LIN; j==rec_bus[i]}
                   -sum{P[i]-P_f[i], i=LIN; j==snd_bus[i]}
                   +sum{Pgen[i]-Pgen_f[i], i=GENTHE; j==bus_genThe[i]}
                   +sum{PgenWin[i]-PgenWin_f[i], i=GENWIN; j==bus_genWin[i]} >= 0)
    @defVar(bl, t[GENTHE] >= 0)
    @addConstraint(bl, t_con1[g=GENTHE],
                    t[g] >= gen_cost_the[g]*Pgen_f[g] +
                    1.2gen_cost_the[g]*(Pgen[g]-Pgen_f[g]))
    @addConstraint(bl, t_con2[g=GENTHE],
                    t[g] >= gen_cost_the[g]*Pgen_f[g])
    @defVar(bl, tw[GENWIN] >= 0)
    @addConstraint(bl, t_w_con1[g=GENWIN],
                    tw[g] >= gen_cost_win[g]*PgenWin_f[g] +
                    1.2gen_cost_win[g]*(PgenWin[g]-PgenWin_f[g]))
    @addConstraint(bl, t_w_con2[g=GENWIN],
                    tw[g] >= gen_cost_win[g]*PgenWin_f[g])

    @setObjective(bl, Min, sum{t[g], g=GENTHE} + sum{tw[g], g=GENWIN})
end
```

On the role of wind covariance estimation in power grid dispatch – a case study using PIPS-IPM

Petra et al., "*Economic Impacts of Wind Covariance Estimation on Power Grid Operations,*" submitted to IEEE Power Systems, 2014.

Case study for the economic dispatch for Illinois grid

- The network consists of 2522 lines, 1908 buses, 870 demand buses, 225 generators, of which 32 are wind farms.
- Wind “installed” capacity is 17%. Adoption in around 15%.

$$
\begin{aligned}
\min_{x, X(\omega), f, F(\omega)} \quad & \sum_{i \in G} c_i x_i + \mathbb{E}_\omega \sum_{i \in G} p_i |x_i - y_i(\omega)| \\
\text{subj.to:} \quad & \tau_n(f) + \sum_{i \in T(n)} x_i = d_n, \forall n \in N \\
& \tau_n(F(\omega)) + \sum_{i \in T(n)} y_i(\omega) = d_n, \forall n \in N, \omega \in \Omega \\
& f, F(\omega) \in U, \forall \omega \in \Omega \\
& x_i, y_i(\omega) \in U_i, \forall i \in G, \omega \in \Omega
\end{aligned}
$$

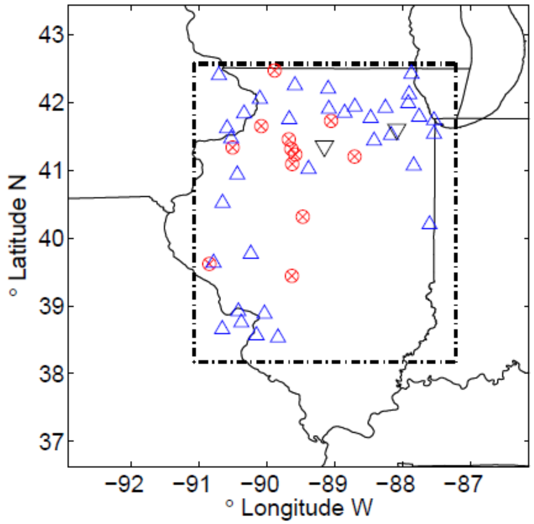
- RBLW covariance matrix (“corr.”) vs diagonal covariance matrix (“indep.”)

- Argonne’s BG/P and BG/Q platforms used in numerical simulations.

Integrating wind samples in economic dispatch models

- Numerical weather forecasting is used to sample wind.

- **Approach 1**: Wind farms bid energy based on their own, independent forecasts. The ISO then considers all the scenarios in the ED model.
  - Correlation among wind farms is lost
  - An exhaustive list of scenarios leads to a gigantic ED problem. Not clear how to bundle scenarios to reduce dimensionality.
- **Approach 2**: Centralized forecast at the ISO level



- Here we show that Approach 2 should be considered: ignoring or missing correlation information leads to inefficient dispatch.

Motivating example – role of correlation in dispatch

- A very simplistic model: 3 generators (of which 2 wind farms and 1 thermal), 1 demand node, no line constraints
- Power outputs of the wind farms are $W_1 \sim \mathcal{N}(w_1, \sigma_1)$ and $W_2 \sim \mathcal{N}(w_2, \sigma_2)$, and the correlation is ρ ($\rho = \mathbb{E}[(W_1 - w_1)(W_2 - w_2)]/(\sigma_1 \sigma_2)$).

- **How does correlation affect the optimal dispatch cost?**
- The optimization problem can be solved analytically, and the (expected) optimal dispatch cost is:

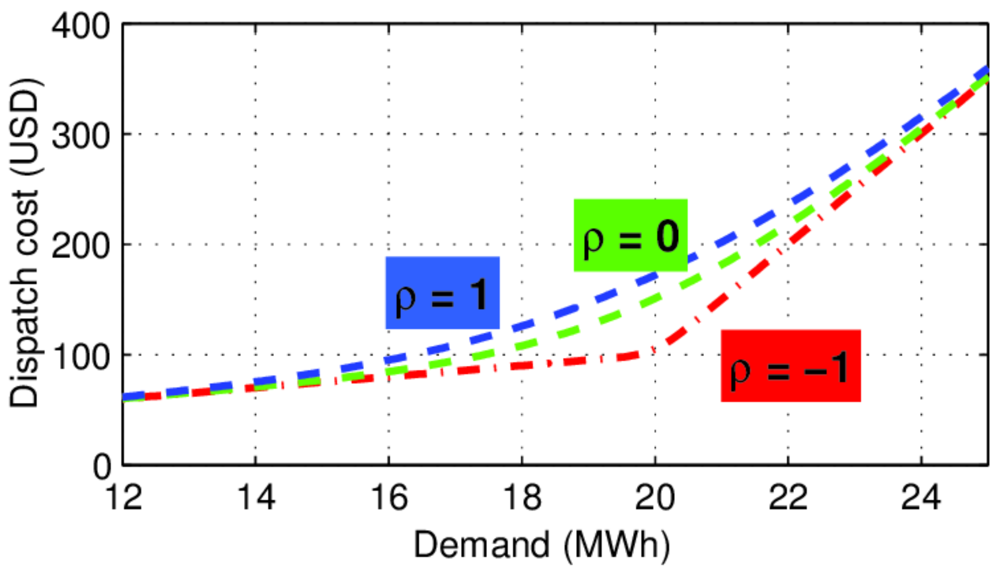
$$c_d(\rho) = c_w d + (c_{th} - c_w)((d - w_1 - w_2)\Phi(d, \sigma_1^2 + 2\rho\sigma_1\sigma_2 + \sigma_2^2) + \sigma^2\phi(d, \sigma_1^2 + 2\rho\sigma_1\sigma_2 + \sigma_2^2))$$

- Here Φ and ϕ are the cumulative distribution and probability distribution functions of $W = W_1 + W_2$

- The optimal dispatch cost is an **increasing function** of the correlation ρ!

Motivating example - continued

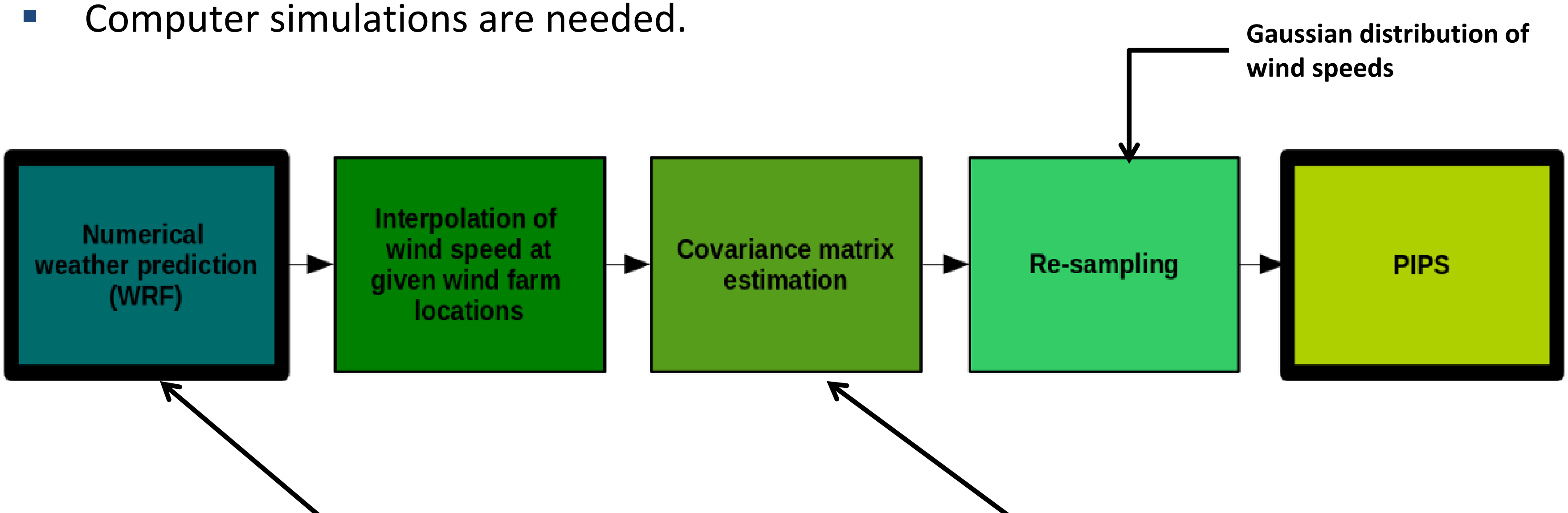
- Ignoring positive correlation leads to an “optimistic dispatch”
- Ignoring negative correlations results in an “pessimistic dispatch

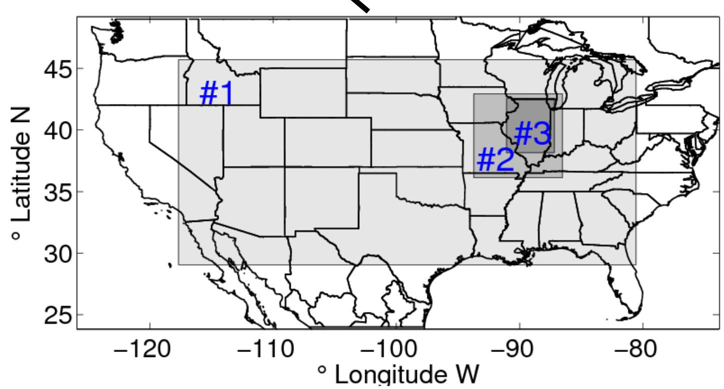


- In both cases higher dispatch costs are obtained over time:
  - “optimistic”: wind predicted more than wind realized: expensive reserves were used.
  - “pessimistic”: wind predicted less than wind realized: more (expensive) thermal power than necessary were dispatched.

What about real-world large-scale power grid systems?

- Analytical analysis of such complex systems is virtually impossible.
- Computer simulations are needed.





Weather forecasting @ Argonne
HPC simulation – 30 samples in RT
(E. Constantinescu)

$$S_\Theta = \rho_1 \cdot I + \rho_2 \cdot S$$

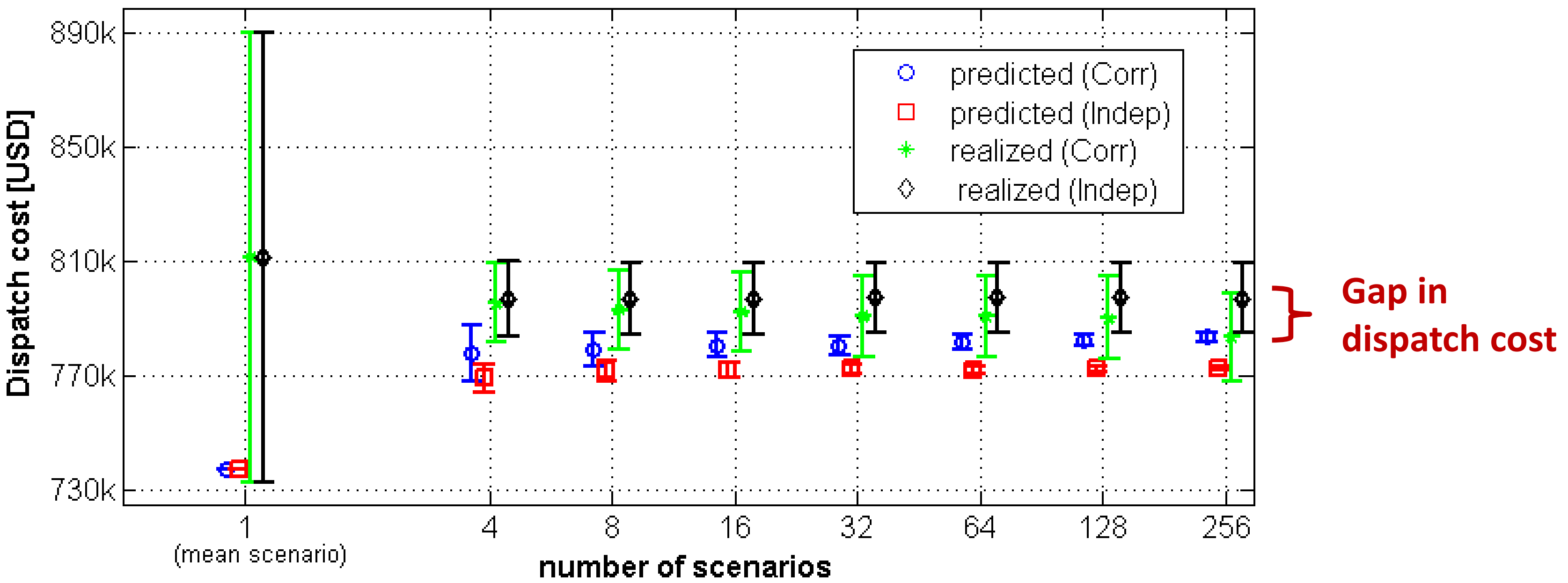
$$\min_{\rho_1,\rho_2} \mathbb{E}\left[\|Q - S_\Theta\|\right]$$

$$\widehat{\Sigma}_{RBLW} = \rho_{RBLW} \cdot I_{p\times p} + (1 - \rho_{RBLW}) \cdot S, \text{ where}$$

$$\rho_{RBLW} = \min\left(\frac{\frac{n-2}{n}\cdot \operatorname{tr}\left(S^2\right) + \operatorname{tr}^2(S)}{(n+2)\cdot\left[\operatorname{tr}(S^2) - \frac{\operatorname{tr}^2(S)}{p}\right]}, 1\right)$$

Dispatch cost – correlation vs independent resampling

95% confidence intervals for the dispatch cost for predicted and realized costs, each with (Corr) and without (Indep) correlation information

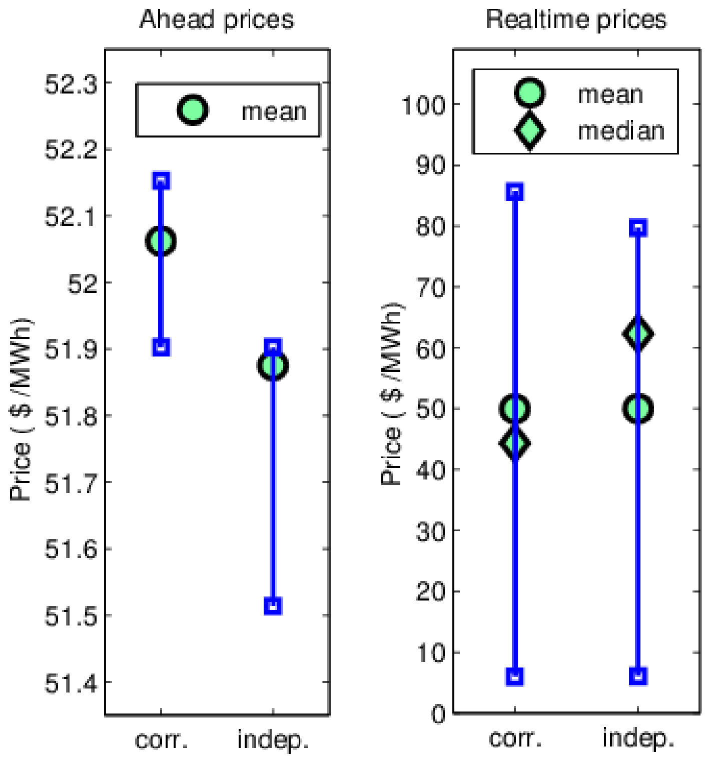


- 1.42% gap or $10,967 (256 scenarios)
- Gap can potentially add up to approx. $100 million over a year.

Prices - correlation vs independent resampling

95% confidence intervals for prices at a typical bus

Gap in prices



- Gap also present in the ahead prices.
- Opportunities for market arbitrage for players with better covariance information.

Thank you for your attention!

Questions?